武漢理工大學

(申请理学硕士学位论文)

微粒群算法的若干改进及应用

培 养 单 位：理 学 院

学 科 专 业：应 用 数 学

研　究　生：熊　　鹰

指 导 老 师：周 树 民　教授

2006 年 11 月

分类号_____ 密　级_____

UDC _____ 学校代码 10497

学　　位　　论　　文

题　目 微粒群算法的若干改进及应用

英　文
题　目 Some Improvements and Applications of Particle Swarm Optimization

研究生姓名 熊　鹰

指导教师 姓名 周树民 职称 教授 学位 硕士

单位名称 理学院 邮编 430070

副指导教师 姓名_____ 职称_____

单位名称_____ 邮编_____

申请学位级别 硕士 学科专业名 应用数学

论文提交日期 2006 年 10 月 论文答辩日期 2006 年 11 月

学位授予单位 武汉理工大学 学位授予日期_____

答辩委员会主席_____ 评阅人_____

_____

2006 年 12 月

摘　要

20 世纪 80 年代，群体智能算法作为一种新兴的演化计算技术已成为越来越多研究者关注的焦点，群体智能的概念源于对蜜蜂、蚂蚁、大雁等群居生物群体行为的观察和研究。通常将这样一种模拟群居性生物中的集体智能行为的智能计算或优化方法称为群体智能算法。微粒群优化算法是一种新型的群体智能算法，源于对鸟群捕食行为的研究，与遗传算法类似是一种基于迭代的优化技术。系统初始化为一组随机解，通过迭代搜寻最优值。目前微粒群算法已广泛应用于函数优化、神经网络训练、数据挖掘、模糊系统控制以及其他的应用领域。

本文从微粒群算法的三种模型出发，在此基础上对其进行了若干改进，并将这些改进用于函数优化、约束优化、整数规划和交叉规划。具体工作如下：(1) 从信息交换方式的角度出发提出了基于收缩因子的自身最好位置赋权微粒群算法，新算法使微粒可以利用更多其他微粒的有用信息，即通过个体极值加权来平衡算法搜索效率和精度之间的矛盾，并改变了微粒的行为方式。(2) 提出了针对多峰函数的避免微粒群陷入局部最优的含步长加速变异算子的微粒群算法及其一种变体，并给出了变异时机和变异概率的详细分析。(3) 针对约束优化问题提出了保证微粒在可行域内运动的混合微粒群算法，并提出了三种初始微粒群的构造方法。(4) 针对含有约束的整数规划问题，提出了一种保证微粒在可行域内运动的整数规划微粒群算法及其改进，该算法及改进算法可以有效地求解约束线性和非线性整数规划。(5) 证明了线性交叉规划的均衡解必存在于两个约束域的边界的交集上的结论并根据此结论提出了线性交叉规划的顶点搜索法。(6) 提出了线性交叉规划的对偶罚函数法，将线性交叉规划转化为非线性规划，并证明线性交叉规划的均衡解可以从非线性规划的最优解中得到。(7) 从天平中得到启示，提出了交叉规划的均衡迭代算法。(8) 提出了基于交叉规划的两种混合微粒群算法，分别用于求解线性交叉规划和非线性交叉规划。

关键词：微粒群算法，约束优化，整数规划，交叉规划

Abstract

In 1980s,Swarm Intelligence Algorithms,a new technology of evolutionary computation, has become the focus of attention to more and more pursuer. The conception of swarm Intelligence originates from observation and investigation into behavior of gregarious colony such as bee, ant and wide goose. Usually, we call the intelligence computation or optimization method that simulates swarm intelligence behavior of gregarious colony as Swarm Intelligence Algorithms. Particle Swarm Optimization, a new Swarm Intelligence Algorithms, originates from the investigation into behavior of bird swarm prey. It is an optimization technology based on iteration as Genetic Algorithm. System is initialized by a group of random solution and search optimization value by iteration. Recently, Particle Swarm Optimization is applied into function optimization, Neural Networks, data mining, Fuzzy Control System and other application field.

The paper begins from the three models of Particle Swarm Optimization and do many improvements. We apply these improvements into Function Optimization, Constrained Optimization, Integer Programming and Interactive Programming. The detailed jobs as follows:(1) The paper put forward an individual best position weight Particle Swarm Optimization based on shrinking gene from the angle of information exchange manner. The new algorithm make particle utilize more available information of other particle. Moreover, it balances the contradiction between efficiency and precision of algorithm search by weight of individual best value and change action mode of particle.(2) The paper put forward a Particle Swarm Optimization and anamorphosis with step-accelerating mutation operator which avoid for particle swarm to plunge into local optimization for multiple hump function, at the same time, we give the detailed analysis of mutation occasion and mutation probability.(3) The paper put forward a hybrid Particle Swarm Optimization which keeps particle swarm acting in feasible region for Constrained Optimization and three methods which construct initial particle swarm.(4)The paper put forward a Particle Swarm Optimization and improving algorithm which keeps particle swarm acting in feasible region for Constrained Integer Programming. The algorithm and improving algorithm can resolve Linear Constrained Integer Programming and Non-linear Constrained

Integer Programming effectively. (5)The paper prove the conclusion that equilibrium solution of Linear Interactive Programming exist in intersection between boundary of two constrained regions .By the conclusion, we put forward vertex-searching method for Linear Interaction Programming.(6) The paper put forward Dual Penalty Function Method for Linear Interactive Programming. It translates Linear Interactive Programming into Non-linear Programming and proves equilibrium solution of Linear Interactive Programming can get from the Optimal Solutions of Non-linear Programming. (7)The paper put forward equilibrium iterative method for Interaction Programming with inspiration of scale.(8)The paper put forward two hybrid Particle Swarm Optimization based on Interaction Programming. The two algorithms resolve the Linear Interaction Programming and Non-linear Interaction Programming.

keywords: Particle Swarm Optimization, Constrained Optimization, Integer Programming, Interactive Programming

目 录

第一章 绪论……..1
1.1 问题的提出……1
1.2 微粒群算法的研究现状……..……3
1.2.1 微粒群算法的发展……..……3
1.2.2 微粒群算法的应用……..……4
1.3 本文的组织……..……5
第二章 微粒群算法的三种典型模型……..……6
2.1 基本 PSO 模型……6
2.2 引入惯性权重的 PSO 模型……7
2.3 带收缩因子的 PSO 模型……8
第三章 微粒群算法的改进……..……9
3.1 自身最好位置赋权 PSO 算法……9
3.1.1 微粒群算法信息交换方式及分析……..……9
3.1.2 基于收缩因子的自身最好位置赋权 PSO 算法……10
3.1.3 几种权重的确定方法……11
3.1.3.1 标准化赋权法……..…11
3.1.3.2 锦标赛赋权法……..……11
3.1.3.3 层次分析法……..12
3.1.4 算法测试及分析……12
3.2 避免陷入局部最优的 PSO 算法……..……13
3.2.1 变异时机和变异概率的分析……14
3.2.1.1 通过微粒的聚集程度来确定变异时机……..14
3.2.1.2 通过适应度的变化率来确定变异时机……..…14
3.2.1.3 通过微粒无进化的次数来确定变异时机……..……14
3.2.1.4 变异概率分析……..……16
3.2.2 含步长加速变异算子的微粒群算法……16
3.2.2.1 探测移动……..……16
3.2.2.2 模式搜索……..……17
3.2.2.3 算法测试及分析……..…18
3.2.2.4 全局最优位置的步长加速变异微粒群算法……..…19
第四章 微粒群算法在约束优化和整数规划中的应用……22
4.1 保证微粒在可行域内运动的微粒群算法……22
4.1.1 初始微粒群的构造……..……22
4.1.1.1 随机压缩半径构造初始可行微粒群……..……22
4.1.1.2 比例压缩半径构造初始可行微粒群……..……23
4.1.1.3 混合压缩半径构造初始可行微粒群……..……24
4.1.2 保证微粒在可行域内运动……..……25
4.1.2.1 随机惯性权重微粒群算法……..…26
4.1.2.2 可行域内迭代的随机惯性权重微粒群算法……..…27
4.1.3 算法测试及分析……..…28
4.2 整数规划的微粒群算法……31
4.2.1 初始微粒群的构造……31

4.2.2 整数规划的微粒群算法……32
4.2.3 算法测试与分析……33
4.2.4 算法的改进……35
第五章 交叉规划的混合微粒群算法……38
5.1 交叉规划及经济背景分析……38
5.2 交叉规划模型……40
5.3 线性交叉规划的顶点搜索法……41
5.4 线性交叉规划的对偶罚函数法……44
5.5 交叉规划的均衡迭代算法……46
5.5.1 天平平衡的启示……46
5.5.2 交叉规划的均衡迭代算法……47
5.5.3 算法测试及分析……48
5.6 交叉规划的混合微粒群算法……49
5.6.1 二人交叉规划模型的转化模型……49
5.6.2 交叉规划的微粒群算法……51
5.6.3 实例及分析……52
第六章 结论与展望……55
致 谢……56
参考文献……57
附录……60

第一章 绪论

1.1 问题的提出

20 世纪 50 年代中期创立了仿生学，人们从生物进化的机理中是受到启发，提出了许多用以解决复杂优化问题的新方法——进化算法，如遗传算法(Genetic Algorithm，GA)[1]、遗传程序设计(Genetic Programming，GP)[2]、进化策略(Evolution Strategies，ES)[3]以及进化规划(Evolutionary Programming，EP)[4]等。虽然这几种方法在实现手段上各有特点、互不相同，但它们所遵循的进化原则是一致的。它们主要是模仿生物学中进化和遗传过程，遵循达尔文的“适者生存、优胜劣汰”的竞争原则，从一组随机生成的初始可行解群体出发，借助复制、重组以及突变等遗传操作，在搜索过程中自动获取并积累解空间的有关知识，逐步向问题的最优解逼近。因此从实质上来说，进化算法是一类具有自适应调节功能的搜索寻优技术，目前它已经被广泛地应用到组合优化问题、机器学习、人工生命、自动控制以及动态系统的故障诊断等领域中。

20 世纪 80 年代，群智能算法作为一种新兴的演化计算技术已成为越来越多研究者关注的焦点，它与 ES 和 GA 有着极为特殊的关系。在没有集中控制且不提供全局模型的前提下，群体智能为寻找复杂的分布式问题的解决方案提供了基础。群智能的概念源于对蜜蜂、蚂蚁、大雁等群居生物群体行为的观察和研究。通常将这样一种模拟群居性生物中的集体智能行为的智能计算或优化方法称为群体智能算法。严格来讲，群体智能是一种在自然界生物群体所表现出的智能现象启发下提出的人工智能模式，是对简单生物群体的智能现象的具体模式研究，即“简单智能的主体通过合作表现出复杂智能行为的特性”。该智能模式需要以相当数目的智能个体来实现对某类问题的求解功能。作为智能个体本身，在没有得到智能群体的总体信息反馈时，它在解空间中的运动方式是完全没有规律的。只有在受到整个智能群体在解空间中运动效果的影响之后，智能个体在解空间中才能体现出具有总体合理寻优特征的运动模式。

最早关于群体智能的研究是 Craig Reynolds 在 1986 年所提出的一个用于模拟鸟类聚集飞行行为的仿真模型 Boid，该模型通过对现实世界中这些群体运动的观察，在计算机中复制和重建了这些运动轨迹，并实现了对这些运动进行抽象建模，以发现新的运动模式。意大利学者 Colorni A、Dorigo M 和 Maniezzo V 于

1992 年首先提出了蚁群算法[5]，它是对蚂蚁群体采集食物过程的模拟，已成功用于许多离散优化问题。

Millonas 在 1994 年提出了群体智能应该遵循的五条基本原则：(1) 相似性原则（Proximity Principle）：群体能够进行简单相似的空间和时间计算；(2) 品质响应原则（Quality Principle）：群体能够对环境中的各类品质因子作出响应；(3) 多样性反应原则（Principle of Diverse Response）：群体的行动和响应范围不应太窄；(4) 稳定性原则（Stability Principle）：群体不应在每次环境变化时都改变自身的行为；(5) 适应性原则（Adaptability Principle）：在能够接受的计算代价内，群体必须能够在适当的时候合理改变自身的行为。以上原则说明，实现群体智能的智能个体必须能够在环境中表现自主性、反应性、学习性和自适应性等智能特性。但是，这并不代表群体中的每个个体都相当复杂，事实恰恰相反，群体智能的核心就是由众多简单个体组成的，群体能够通过相互之间的简单合作来实现某一较复杂的功能，完成某一较复杂的任务。其中，“ 简单个体 ” 是指只具有简单能力或智能的单个个体，而“ 简单合作 ” 是指个体与其邻近个体进行某种简单的直接通讯或通过改变环境因素间接与其他个体通讯，从而实现相互影响和协同合作[6]。

微粒群优化算法 (Particle Swarm Optimization，PSO) 是一种新型的群体智能算法，由 Eberhart R.C 博士和 Kennedy J 博士于 1995 年提出[7]。PSO 源于对鸟群捕食行为的研究，与 GA 类似是一种基于迭代的优化技术。系统初始化为一组随机解，通过迭代搜寻最优值。目前已广泛应用于函数优化、神经网络训练、数据挖掘、模糊系统控制以及其他的应用领域。

PSO 是模拟鸟群的捕食行为。设想这样一个场景：一群鸟在随机搜寻食物。在这个区域里只有一块食物。所有的鸟都不知道食物在哪里。但是他们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢？最简单有效的就是搜寻目前离食物最近的鸟的周围区域。PSO 从这种模型中得到启示并用于解决优化问题。在 PSO 中每个优化问题的解都是搜索空间中的一只鸟，我们称之为“ 微粒 ”。所有的微粒都有一个由被优化的函数决定的适应值，每个微粒还有一个速度，决定它的方向和位置。然后微粒就追随当前的最优微粒在解空间中搜索。PSO 初始化为一群随机微粒（随机解），然后通过迭代找到最优解。在每一次迭代中，

微粒通过跟踪两个“极值”来更新自己。第一个就是微粒本身所找到的最优解，这个解叫做个体极值 pBest，另一个极值是整个种群目前找到的最优解，这个极值是全局极值 gBest。此外也可以不用整个种群而只用其中一部分作为微粒的邻居，那么所有邻居中的极值就是局部极值，微粒始终跟随这两个极值变更自己的位置和速度直到找到最优解。

我国学者李晓磊等于 2002 年根据鱼的特性提出了鱼群算法[8](Fish Swarm,FS)，也是一种群体智能算法，它是一种基于鱼类行为的寻求全局最优的新思想，是行为主义人工智能的又一典型应用。

1.2 微粒群算法的研究现状

PSO 自提出以来引起了国际上相关领域众多学者的关注和研究，成为国际演化计算界研究的热点。

1.2.1 微粒群算法的发展

在 PSO 的改进方面，首先是由 Kennedy 和 Eberhart 在 1997 年提出的二进制 PSO[9]，该方法可用于神经网络的结构优化；其次，为了提高算法的收敛性，Shi 和 Eberhart 于 1998 年对 PSO 的速度项引入了惯性权重，并提出在进化过程中动态调整惯性权重以平衡收敛的全局性和收敛速度，该进化方程已被相关学者称为标准微粒群算法[10]。2001 年，Shi 又提出了自适应模糊调节惯性权重的 PSO，在对单峰函数的处理中取得了良好的效果，但无法推广。Clerc 于 1999 年在进化方程中引入收缩因子以保证算法的收敛性，同时使得速度的限制放松[11]。有关学者已通过代数方法对此进行了详细的算法分析，并提出了参数选择的指导性建议，同年，Angeline 借鉴进化计算中的选择概念，将其引入 PSO 中，通过比较各微粒的适应值淘汰差的微粒，而将具有高适应值的微粒进行复制以产生等额的微粒来提高算法的收敛性。而 Lovbjerg 等人进一步将进化计算机制应用于 PSO，如复制，交叉等，给出了交叉的具体形式，通过仿真实验说明了算法的有效性。Bergh 通过使微粒群中的最佳微粒始终处于动态状态，得到了保证收敛到局部最优的 GCPSO，但其性能不佳[12]。为了提高算法收敛的全局性，防止微粒陷入“早熟”，保证微粒的多样性是其关键。Suganthan 在标准 PSO 中引入了空间邻域的概念，将处于同一空间邻域的微粒构成一个子微粒群分别进化，并随着进化动态地改变选择阈值以保证群体的多样性；Kennedy 引入邻域拓扑的概念来调整增加

邻域间的信息交流，提高群体的多样性。Lovbjerg 等人于 2001 年将遗传算法中的子群体概念引入 PSO 中，同时引入繁殖算子以进行子群体的信息交流。Bergh 又于 2001 年提出了协同 PSO[13]，其基本思想是用 N 个相互独立的微粒群分别在 D 维的目标搜索空间中的不同维方向上进行搜索。高鹰等于 2004 年提出了基于模拟退火的 PSO[14]和免疫 PSO[15]。Higashi 等人分别提出了自己的变异 PSO 算法，基本思路均是希望通过引入变异算子跳出局部极值的吸引，从而提高算法的全局搜索能力，得到较高的搜索成功率。除以上的 PSO 外，还出现了量子 PSO、耗散 PSO、自适应 PSO 等混合改进算法，也有采取 PSO 与基于梯度的优化方法相结合的 PSO。

1.2.2 微粒群算法的应用

PSO 算法应用面很广泛，Kassabalidis I 等利用 PSO 实现对卫星无限网络路由的自适应调整，提高网络容量的有效利用率。Robinson J，Sinton S 和 Rahmat-Samii 将 PSO 用于 PCHA(剖面波状喇叭天线)的优化设计，并与 GA 的优化效果进行了比较，在此基础上又研究了二者混合应用的可行性。其中所完成的一系列实验证实 PSO 作为一种新型的优化算法具备解决复杂工程优化问题的能力。Ismail A,Engelbrecht AP 利用 PSO 实现了对人工神经网络权值和网络模型结构的优化，并将研究结果应用于“自然语言组词”的分析方法设计。Eberhart R.C 与 Hu X 采用经 PSO 优化的神经网络实现了对人类肢体颤抖现象的分析，并完成了对正常颤抖和帕金森氏症的诊断功能。在疾病和乳腺肿瘤是良性和恶性的判断、心脏病的诊断，PSO 训练的神经网络也取得了较高的诊断成功率。El Gallad A I,El Hawary,M E Sallam,Kalas 同样采用 PSO 对神经网络进行了优化，并利用其设计了电力变压器的智能保护机制。Yoshida H,Kawata K,Fukuyama Y,Nakanishi 利用 PSO 实现了对各种连续和离散控制变量的优化，从而达到了控制核电机组电流稳定输出电压的目的[16]。

PSO 算法和其他进化算法类似，能用于求解大多数优化问题。比如多元函数的优化问题，包括带约束的优化问题。经过大量的实验研究发现，PSO 算法在解决一些典型优化问题时，能够取得比遗传算法更好的优化结果。PSO 还在动态问题和多目标优化问题中得到应用。

PSO 算法也被成功地应用于电力系统领域。这里主要涉及到带约束条件的，

使用不同版本的 PSO 算法相结合用来决定对连续和离散的控制变量和控制策略的问题。日本 Fuji 电力公司的研究人员将电力企业著名的 RPVC（Reactive Power and Voltage Control）问题简化为函数最小值问题，并使用改进的 PSO 算法进行求解，与传统的方法如专家系统、 敏感性分析相比，实验产生的结果证明了 PSO 算法在解决该问题时的优势。将 PSO 算法与 BP 神经网络算法相结合训练神经网络用于电动汽车燃料电池组实时充电情况的模拟和各种生物化学成分的优化组合进而生成人工合成微生物的过程。

除了以上领域外，PSO 在自动目标检测、生物信号识别、决策调度、系统辨识以及游戏训练、分类、信号处理、机器人应用等方面也取得了一定的成果。目前在模糊控制器设计、车间作业调度、机器人实时路径规划、自动目标检测、语音识别、烧伤诊断、探测移动目标、时频分析和图象分割等方面已经有成功应用的先例。

1.3 本文的组织

本文章节组织如下：第一章首先对微粒群算法进行了综述，分别从微粒群算法的提出、微粒群算法的发展和微粒群算法的应用三个方面进行了详细地说明。第二章简单介绍了微粒群算法的三种典型模型，这也是后面章节的基础。第三章提出了两种改进的微粒群算法，即基于收缩因子的自身最好位置赋权 PSO 算法和含步长加速变异算子的微粒群算法，分别用于高维单峰函数和多峰函数的优化。第四章第一部分提出了一种保证微粒在可行域内运动的求解约束优化问题的微粒群算法，第二部分提出了一种求解整数规划问题的微粒群算法及其改进。第五章首先介绍交叉规划的经济背景，然后介绍交叉规划的几种模型并给出了将交叉规划转化为二层规划的一种转化模型，最后提出了四种求解交叉规划的算法，即线性交叉规划的顶点搜索法、线性交叉规划的对偶罚函数法、交叉规划的均衡迭代算法和交叉规划的混合微粒群算法，并对这四种算法的优缺点进行了对比分析。第六章是总结与展望。

第二章　微粒群算法的三种典型模型

自 1995 年微粒群算法问世以来，不同领域的研究人员曾提出各种算法模型，从不同角度对微粒群算法进行了详细的分析、设计。其中倍受推崇的是 Kenney、Eberhart、Yuhui Shi 和 Clerc，他们依据微粒群算法的模拟思想，分别构造了基本 PSO 模型，引入惯性权重的 PSO 模型和引入收缩因子的 PSO 模型，并通过大量的实验研究，详细分析了模型中不同控制参数的意义与作用，并为其确定了相应的参考取值。这三种算法模型被后来的研究人员视为微粒群算法的经典及研究基础。本文对 PSO 的改进都是基于这三种模型。

2.1 基本 PSO 模型

微粒群算法是由 Kennedy 和 Eberhart 等于 1995 年开发的一种演化计算技术，来源于对一个简化社会模型的模拟。PSO 算法与遗传算法类似，也是一种群体智能算法，根据对环境的适应度将群体中的个体移动到好的区域，然而它不像遗传算法那样对个体使用演化算子，而是将每个个体看作 D 维空间中一个没有体积和质量的微粒，在搜索空间中以一定的速度飞行，该飞行速度根据个体的飞行经验和群体的飞行经验进行动态调整。假设一个包含 m 个微粒的微粒群在 D 维空间飞行，设 $X_i = (x_{i1}, x_{i2}, \mathrm{L}\ , x_{id})$ 为微粒 i 的当前位置，$V_i = (v_{i1}, v_{i2}, \mathrm{L}\ , v_{id})$ 为微粒 i 的当前飞行速度，$P_i = (p_{i1}, p_{i2}, \mathrm{L}\ , p_{id})$ 为微粒 i 所经历的最好位置，也就是微粒 i 所经历过的具有最好适应值的位置，称这个位置为自身最好位置；$p_g = (p_{g1}, p_{g2}, \mathrm{L}\ , p_{gd})$ 为群体所有微粒经历的最好位置。有了上面的定义，对每一代，基本微粒群算法的进化方程可描述为：

$$v_{id}(t+1) = v_{id}(t) + c_1 r_{1d}(t)(p_{id}(t) - x_{id}(t)) + c_2 r_{2d}(t)(p_{gd}(t) - x_{id}(t))$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

其中下标 $d(1 \le d \le D)$ 表示微粒的第 d 维，i 表示微粒 i，t 表示第 t 代，c_1 为自身加速常数，c_2 为全局加速常数，c_1, c_2 一般在 $0 \sim 2$ 间取值，$r_1 \sim U(0,1)$，$r_2 \sim U(0,1)$ 为两个相互独立的随机变量，$v_{id}(t) \in [-v_{\max}, v_{\max}]$，$x_{id}(t) \in [-x_{\max}, x_{\max}]$，如果 $v_{id}(t)$、$x_{id}(t)$ 超出边界值，就用边界值取代 $v_{id}(t)$、$x_{id}(t)$，$v_{\max}$、$x_{\max}$ 是依不同的目标函

数和不同的搜索空间而不同的常数。

基本 PSO 算法的实现过程如下所述：首先，确定 m 、c_1、c_2 的值，然后将 m 个微粒均匀地散布到搜索空间，即设置 $x_i(i=1,2,\mathrm{L}\ ,m)$ 的初值，使这些初值在搜索空间的位置均匀分布；再设置每一个微粒的速度 $v_i(i=1,2,\mathrm{L}\ ,m)$ 为 $[-v_{\max},v_{\max}]$ 上的随机数，这样就完成了初始化的工作，下一步进入迭代循环，在规定的循环次数内，依迭代公式确定每一个微粒的搜索空间的下一个位置，并依据优化的目标函数求得每一个微粒的历史最优位置 p_i，以及微粒群的所有 p_i 中的最优位置 p_g，直到求得满足条件的最优解和最优值。

2.2 引入惯性权重的 PSO 模型

从基本微粒群算法模型可以看出，微粒的飞行速度相当于搜索步长，其大小直接影响着算法的全局收敛性。当微粒的飞行速度过大时，能够保证各微粒以较快的速度飞向全局最优解所在的区域。但是当逼近最优解时，由于微粒的飞行速度缺乏有效的控制与约束，则将很容易飞越最优解，转而去探索其他区域，从而使算法很难收敛于全局最优解。这一现象同时也说明了算法在速度缺乏有效控制策略时不具备较强的精细搜索能力。

为了有效地控制微粒飞行的速度，使得算法能够达到全局探索与局部开挖功能间的有效平衡，单靠基本微粒群算法模型中施加的 $V_{\max}$ 是不够的，Eberhart 和 Yuhui Shi 在基本微粒群算法模型中引入了惯性权重系数 w，以实现对微粒飞行速度的有效控制与调整。这一思想源于模拟退火算法，w 类似于模拟退火中的温度控制参数。具体的计算模型如下：

$$v_{id}(t+1)=wv_{id}(t)+c_1r_{1d}(t)(p_{id}(t)-x_{id}(t))+c_2r_{2d}(t)(p_{gd}(t)-x_{id}(t))$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$$

从上式易知，w 越大，微粒的飞行速度越大，微粒将以较大的步长进行全局搜索；w 越小，微粒的速度步长越小，微粒将趋向于进行精细的局部搜索。因此，在搜索过程中对 w 进行动态调整：在算法开始时，可以给 w 赋予一个较大正值，随着搜索的进行，可以线性地使 w 逐渐减少，这样可以保证在算法开始时，各微粒能

够以较大的速度步长在全局范围内搜索到较好的种子，而在搜索后期，较小的 w 值则保证微粒能够在极点周围作精细地搜索，从而使算法以较大的概率以一定精度收敛于全局最优解。设 w 从 b 到 a 随迭代次数线性减少，则 $w(t)=b-\frac{t}{n}(b-a)$，其中 n 为最大截止代数，这样惯性权重 w 看作迭代次数的函数。Eberhart 和 Yuhui Shi 设计了一系列实验认为 $b=0.9, a=0.4$ 时算法性能较好[17]。

2.3 带收缩因子的 PSO 模型

为了有效地控制微粒的飞行速度，使算法达到全局探测与局部开挖两者间的有效平衡，Clerc 构造了引入收缩因子的 PSO 模型。模型如下

$$v_{id}(t+1)=K(v_{id}(t)+c_1r_{1d}(t)(p_{id}(t)-x_{id}(t))+c_2r_{2d}(t)(p_{gd}(t)-x_{id}(t)))$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$$

其中，K 称为收缩因子，$K=\frac{2}{\left|2-l-\sqrt{l^2-4l}\right|}$，且 $l=c_1+c_2, l>4$，一般地 $c_1=c_2=2.05$。此模型中 K 的作用类似于参数 $V_{\max}$ 的作用，用来控制与约束微粒的飞行速度。但实验结果表明，K 比 $V_{\max}$ 更能有效地控制微粒速度的振动。Eberhart 和 Yuhui Shi 详细分析比较了 w 与 K 两种参数对 PSO 算法性能的影响，他们谦虚地承认 Clerc 所提出的收缩因子 K 比惯性权重 w 更能有效地控制微粒的飞行速度，同时增强了算法的局部搜索能力。

第三章 微粒群算法的改进

自微粒群优化算法提出以来，由于它的计算快速性和算法本身的易实现性，引起了国际上相关领域众多学者的关注和研究。但 PSO 算法也存在很多缺点，如对于性能不好的高维函数收敛精度低和不易收敛，另外，PSO 算法易于陷入局部最优点。本章就这两大缺点提出了两种改进的 PSO 算法，前者加快算法收敛到全局最优点，后者通过引入变异算子避开局部最优点。

3.1 自身最好位置赋权 PSO 算法

3.1.1 微粒群算法信息交换方式及分析

微粒群算法是一种群体智能算法。微粒群可以认为是微粒在 D 维空间内，按一定规律传递信息，并根据信息的变化改变自身状态所产生的自组织行为。微粒群的信息主要来自由各微粒的个体极值构成的矩阵 $p=(p_1,p_2,\mathrm{L}\ ,p_n)^T$，微粒群算法中微粒从 p 中提取的信息有群体最优位置 p_g 和每个微粒自身经历的个体最优位置 p_i。群体最优位置使得微粒能够快速收敛形成微粒群，并对全局极值的邻域进行搜索；个体自身经历最优位置保证微粒不至于过快收敛到群最优而陷入局部最优点，使得微粒能够在一次迭代中对个体极值和全局极值之间的区域进行搜索。

微粒群算法是一种群搜索算法，它之所以有高效的搜索性能，是因为群体的合作。正如文献[18]所说：“ 社会行为有两个主要的目的，一是每个个体能够在搜索食物的过程中协助其他群体的成员，二是群体合作能够提高搜索效率。” 换言之，每个微粒能够向群体提供信息并且每个微粒又能够协助其他微粒进行搜索。

每个事物的优缺点是辨证的，微粒群算法的缺点也正是由于其群体搜索的高效性而导致算法易于陷入局部最优。一种简化微粒群算法速度的更新公式为：

$$v_{id}(t+1)=v_{id}(t)+c_2r_{2d}(t)(p_{gd}(t)-x_{id}(t))$$

简化微粒群算法微粒只在群最优邻域进行搜索。因为所有的微粒都参与了群最优的邻域搜索，所以简化微粒群算法是局部搜索最高效率的搜索算法，但也是最易陷入局部最优的算法。基本微粒群算法与简化的微粒群算法相比，提出了个

体极值来延缓微粒群搜索范围的快速收敛而提高搜索精度。但同时也增大了微粒利用的信息量，微粒同时利用了自身的个体极值。

所以微粒群算法的效能取决于微粒获得的信息量与搜索效率和精度的平衡。在基本微粒群算法中，每个微粒获得的信息有个体极值和全局极值两种信息，那么其他微粒的个体极值信息对于微粒运动是否有参考价值呢？如何让微粒获得更多的信息并合理利用以提高算法的性能呢？基于上述这些问题，本节引入了自身最好位置赋权 PSO 算法。

3.1.2 基于收缩因子的自身最好位置赋权 PSO 算法

通过分析微粒群算法的一些特点，可以看出微粒获得信息决定了其行为方式和微粒群的组织方式。社会学家 E · O · Wilson 在文献[19]中说：至少在理论上，在群体搜索食物的过程中，群体中的每个个体可以从群体的新发现和群体中的所有其他个体的经验中受益。同时微粒群算法的创始人 Kennedy 和 Eberhart 在[7]中也指出：种群中的信息分享是一种进化结果，这个假设也是微粒群优化算法发展的基础。本小节也正是继承这种思想提出改进的收缩因子 PSO 模型。

设 $p_i(i=1,2,\mathrm{L}\ ,n)$ 表示微粒 i 的自身最好位置，对每个微粒 i，我们赋予微粒 i 自身最好位置一个权重 ω_i，$0<\omega_i<1$ 且 $\sum_{i=1}^{n}\omega_i=1$，加权得 $\bar{p}=\sum_{i=1}^{n}\omega_i p_i$，用 $\bar{p}$ 代替微粒自身最好位置 $p_i(i=1,2,\mathrm{L}\ ,n)$。则改进的收缩因子 PSO 模型为：

$$v_{id}(t+1)=K(v_{id}(t)+c_1r_{1d}(t)(\bar{p}_d(t)-x_{id}(t))+c_2r_{2d}(t)(p_{gd}(t)-x_{id}(t)))$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$$

其中，K 称为收缩因子，$K=\dfrac{2}{\left|2-l-\sqrt{l^2-4l}\right|}$，且 $l=c_1+c_2, l>4$。

将带收缩因子的 PSO 模型中的 p_{id} 改为 $\bar{p}_d$ 有以下几方面的优势：从信息量角度说，新算法中每个微粒借鉴了其他微粒的经验；从行为方式上讲，微粒不再在群最优和自身最优之间进行搜索，而是在群最优和微粒最优加权中心位置之间进行搜索，改进的收缩因子 PSO 模型不同于单纯的收缩因子 PSO 模型，并且不是所有的微粒都参与群最优的邻域搜索，部分微粒会在个体加权极值和全局极值之

间进行邻域搜索，而且，在收敛过程中，个体加权极值和全局极值的位置不断被微粒搜索接近，使得算法最终收敛。

3.1.3 几种权重的确定方法

权重对自身最好位置赋权 PSO 算法性能的影响是重要的，好的权重可以加快算法收敛的速度和算法收敛的精度，相反，如果权重选得不恰当将会导致微粒进化缓慢甚至停止进化。本节提出了几种确定权重的方法，如标准化赋权法、锦标赛赋权法、层次分析法。

3.1.3.1 标准化赋权法

好的个体最优微粒应该有较大的权重，这样才能保证微粒始终朝好的方向进化，那么如何刻画微粒的好与坏呢？显然个体最优微粒的适应值是反映微粒好坏的标准，对于极小化问题，适应值越小，则它的权重应该越大。基于此，标准化赋权法的权重定义为：

$$a_i = \frac{\max\limits_i f(p_i) + \rho - f(p_i)}{\max\limits_i f(p_i) - \min\limits_i f(p_i)} \tag{3.1}$$

$$\omega_i = \frac{a_i}{a_1 + a_2 + \mathrm{L} + a_n}$$

其中 $f(p_i)$ 表示自身最好微粒 p_i 的适应值，$\rho \geq 0$，当 $\rho = 0$ 时适应值最大的自身最好微粒的权重为 0，这样适应值最大的自身最好微粒的信息没有用到。我们选取适当小的正数 ρ，既保证 $a_i \neq 0$，又使得各 a_i 的值不因为 ρ 的值过大而拉不开距离。

3.1.3.2 锦标赛赋权法

锦标赛赋权重法中的 a_i 不是通过自身最好位置适应值标准化得到的，而是通过自身最好位置适应值的两两比较，如果当前自身最好微粒的适应值不大于某个自身最好位置适应值，则每次授予该自身最好位置一分，对每个个体重复这一过程。算法伪代码如下：

$$
\begin{aligned}
&a_i = 0 \\
&for \;\; i = 1:n \\
&\quad for \;\; j = 1:n \\
&\quad\quad if \;\; f(p_i) \le f(p_j) \\
&\quad\quad\quad a_i = a_i + 1 \\
&\quad\quad end \\
&\quad end \\
&end
\end{aligned}
$$

则 $\omega_i = \dfrac{a_i}{a_1 + a_2 + \mathrm{L} \;\; + a_n}$ 表示 p_i 的权重。

3.1.3.3 层次分析法

层次分析法首先要将每个自身最好位置的适应值通过标准化赋权法中的（3.1）式或通过锦标赛赋权法中的程序代码求出 a_i，其对应的赋权方法分别称为标准化层次分析赋权法和锦标赛层次分析赋权法。然后作矩阵 B

$$
B = \begin{pmatrix} b_{11} & \mathrm{K} & b_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ b_{n1} & \mathrm{L} & b_{nn} \end{pmatrix}
$$

其中 $b_{ij} = \dfrac{a_i}{a_j}(i, j = 1, 2\mathrm{L} \;\; n)$，设 $\lambda = (\lambda_1, \lambda_1, \mathrm{K}\;, \lambda_n)$ 为矩阵 B 的最大特征值对应的特征向量，则自身最好位置 p_i 的权重为：

$$
\omega_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \mathrm{L} \;\; + \lambda_n}
$$

其中 λ_i 表示向量 λ 的第 i 个分量。

3.1.4 算法测试及分析

取微粒群规模 n=20，$c_1 = c_2 = 2.05$，用基于收缩因子的自身最好位置赋权 PSO 算法，分别用标准化赋权法、锦标赛赋权、标准化层次分析赋权法、锦标赛层次分析赋权法以及无权重的带收缩因子 PSO 算法测试 20 维、30 维、40 维、50 维的 rosenbroke 函数，$\rho = 1$，变量每维的取值范围为[-10，10]，截止代数为 2000 代，算法执行 10 次取平均值，平均最优值列于表 3-1。

表 3-1　基于收缩因子的自身最好位置赋权 PSO 算法计算结果

维数	平均最优值				
	无权重	标准化赋权法	标准化层次分析赋权法	锦标赛赋权	锦标赛层次分析赋权法
20	211.5013	8.9444	6.3912	14.9996	13.9189
30	824.2361	24.1160	18.8074	19.9841	25.2413
40	1.1727e+3	62.3223	46.8340	51.4721	42.9519
50	9.6403e+4	82.9601	73.7834	94.0032	94.2899

从表 3-1 中可以看出，带自身最好位置权重的收缩因子 PSO 算法的求解结果比无权重的收缩因子 PSO 的结果要好得多，但不同的赋权方法对计算结果的影响不明显，相对而言，标准化层次分析赋权法的结果最好。

本节从微粒获得信息的角度出发，提出了一种新的改进的带收缩因子的 PSO 算法。新算法使微粒可以利用更多其他微粒的有用信息，即通过个体加权极值来平衡算法搜索效率和精度之间的矛盾，并改变了微粒的行为方式。实验结果表明，改进的基于收缩因子的自身最好位置赋权 PSO 算法在收敛速度方面优于单纯的带收缩因子的 PSO 算法。

3.2 避免陷入局部最优的 PSO 算法

在标准的 PSO 算法搜索的后期，微粒群会向局部最优点或全局最优点收敛，此时，每个微粒的自身最好位置、整个微粒群的全局最好位置和每个微粒的当前位置都会趋于同一点，而每个微粒的运动速度则趋于零，因此微粒群所趋向的那个点即为微粒群算法的最终求解结果的极限值，如果目标函数为非凸函数或多峰函数，PSO 算法的求解结果为局部最优的概率远大于为全局最优的概率，如果不采取一些措施，则微粒群算法对这样的目标函数的求解会陷入局部最优。目前已经提出了许多避免陷入局部最优的 PSO 算法，如局部版微粒群算法[20]，保证种群多样性的微粒群算法[21]，带高斯变异算子的微粒群算法[22]，含维变异算子的微粒群算法[23]，含速度变异算子的微粒群算法[24]。

本节将给出变异时机和变异概率的详细分析，在分析的过程中，提出了一种单个微粒变异的微粒群算法，最后提出了基于步长加速变异的微粒群算法以避免陷入“ 早熟 ”。

3.2.1 变异时机和变异概率的分析

很多参数都可以表现微粒群的收敛程度，它们的大小和变化情况都可以作为确定变异时机的依据，以下我们给出几种变异时机的参数选择。

3.2.1.1 通过微粒的聚集程度来确定变异时机

用一个参数 $Lcon$ 来代表微粒群当前位置的质心到各个微粒的当前位置的距离之和，即

$$Lcon = \sum_{i=1}^{n} d(\overline{x}, x_i)$$

其中 n 表示微粒的个数，x_i 为第 i 个微粒的当前位置，$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ 表示微粒群当前位置的质心，$d(\overline{x}, x_i)$ 表示微粒 i 到微粒群当前位置的质心的距离。则 $Lcon$ 值越小微粒群的收敛度越大，当 $Lcon$ 为零时，微粒群的收敛度达到最大。再设置一个 $Lcon$ 的临界值，一旦 $Lcon$ 小于该临界值，就进行变异。

3.2.1.2 通过适应度的变化率来确定变异时机

当适应度值的变化率——即本次迭代求得的适应度值与本次迭代之前 M 次迭代时求得的适应度值之差的绝对值，与本次迭代求得的适应度值之比小于某一阈值时，开始变异。这一变异时机可以针对整个微粒群，也可以针对某一单个微粒。如果是针对整个微粒群，则取全局最好位置 p_g 的适应值的变化率作为变异时机的参数选择，当这一变化率小于某一阈值时，整个微粒群以某一种变异方式以一定的变异概率进行变异。如果是针对单个微粒，则以此微粒的适应值的变化率作为变异时机的参数选择。

3.2.1.3 通过微粒无进化的次数来确定变异时机

这一变异时机与适应值的变化率比较类似，也可以针对整个微粒群和单个微粒。对某些高峰函数，微粒群的进化非常缓慢，我们难以找出很好的阈值来评价适应值变化率的变异时机，如果微粒仍在进化时突然变异，则会破坏微粒群的结构。于是我们可以通过微粒无进化的次数作为变异时机来评价整个微粒群的收敛度或个体微粒进化的前景。当微粒群的全局最好位置的适应值连续 M 次都没有

得到改进，则认为微粒群已经聚集到某一局部最优位置，此时整个微粒群以某一种变异方式以一定的变异概率进行变异。对于单个微粒，如果它的适应值连续 M 次都没有得到改进，就可以对它以一定的变异概率和变异方式进行变异。

我们以 Rastrigin 函数作为测试函数，分别测试整个微粒群变异的高斯变异微粒群算法和单个微粒变异的高斯变异微粒群算法各 100 次。

$$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10) \qquad -5.12\le x_i\le 5.12$$

该函数被称为 Rastrigin 函数，它是一个多峰函数，在 $x_i=0(i=1,2,\mathrm{L}\ ,n)$ 时达到全局最小点，在 $S=\{x_i\in(-5.12,5.12),i=1,2,\mathrm{L}\ ,n\}$ 范围内大约存在 $10n$ 个局部极小点。取微粒数为 30，空间的维数为 10，取微粒无进化的次数作为变异时机，对于整个微粒群变异，则微粒的全局最优值连续 10 次都没有得到改进则微粒变异。对于单个微粒变异，则每个微粒连续 10 次没有进化则变异，变异概率随迭代次数的函数关系均为 $p_c(i)=1/(1+\sqrt{i})$，对这两种高斯变异微粒群算法都测试 100 次，当 $fun(p_g)<10^{-10}$ 时，则该次测试结束，记录收敛代数，并将这 100 次的收敛代数从小到大排列，作为纵坐标，横坐标取[1,100]上的整数，如图 3-1。

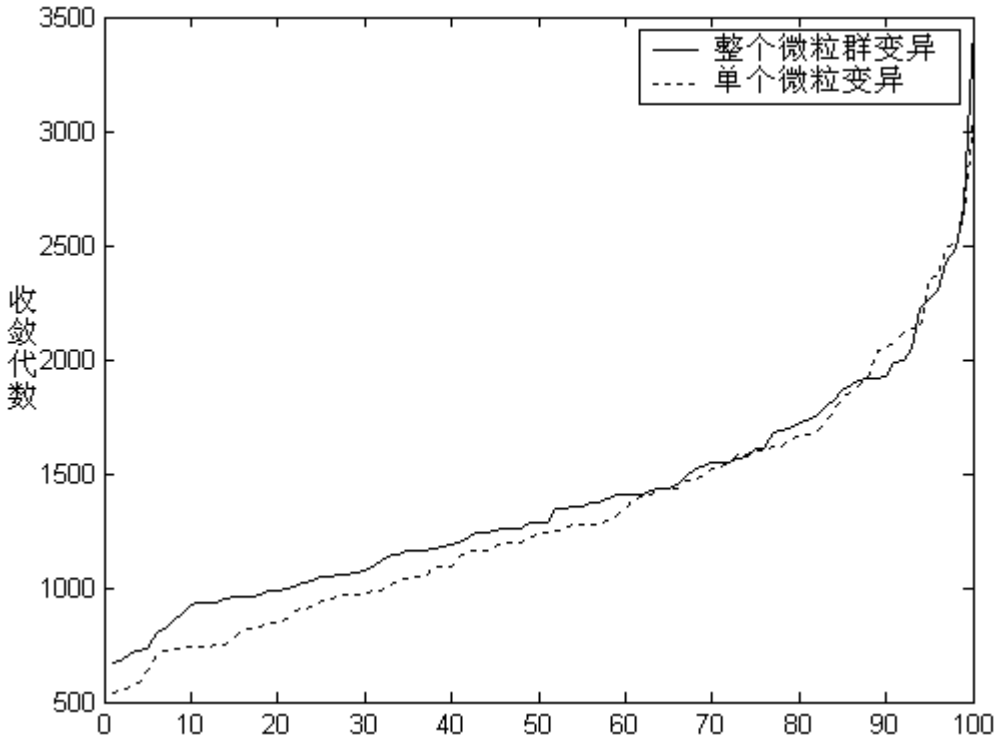

图 3-1 整个微粒群变异和单个微粒变异的高斯变异 PSO 算法收敛代数对比图

图3-1中实线和虚线的纵坐标分别表示整个微粒群变异的高斯变异微粒群算法和单个微粒变异的高斯变异微粒群算法的收敛代数。由于微粒群算法有很大的随机性，所以每次测试结束时收敛代数不相同，我们将两种变异算法各测试的100 次收敛代数从小到大排序，横坐标都取[1,100]上的整数，这样横坐标为$i(i=1,2,\mathrm{L}\ ,100)$时，对应的纵坐标分别表示两种算法100次测试的收敛代数中第i小的收敛代数，因此这样的曲线能够反映两种算法多次测试的收敛水平，在后续章节中涉及到两种或多种算法的比较时，我们还会用到。

从图3-1中明显看出单个微粒变异比整个微粒群变异有更快的收敛速度。

3.2.1.4 变异概率分析

变异概率具有很大的随机性，过大的变异概率会改变微粒群的结构，过小的变异概率又起不到变异的作用。随着微粒群的不断进化，全局最优值不断得到改进，变异次数也应该不断减少，所以变异概率应该随迭代次数不断减少，这就如同标准微粒群算法中的权重w随迭代次数的增加而减少一样，一般地，取变异概率$p_c(t)=\frac{1}{1+t^{\alpha}}$（$0<\alpha<1$，其中$t$为微粒群迭代次数）这一随迭代次数增加非线性单调递减的函数。

3.2.2 含步长加速变异算子的微粒群算法

含步长加速变异算子的微粒群算法是将无约束优化算法中的步长加速法作少许变动，然后作为微粒群算法的一种变异策略，该变异策略与其他变异策略最大的不同之处是可不需要变异时机，可以直接将变异作为微粒群算法的一个算子，在微粒群的每一步迭代中，更新了每个微粒的速度和位置后，以一定的变异概率对每个微粒做步长加速变异。这种改进的微粒群算法在求解多峰函数的优化问题时非常有效。步长加速由两个动作构成，称为探测移动和模式搜索，我们将它们稍作改动然后再组合起来，形成一种新的变异策略。3.2.2.1 和 3.2.2.2 主要参考了文献[25]。

3.2.2.1 探测移动

设n为空间的维数，对每个微粒x_i，沿着该微粒的n个维的正反方向以λ（由于在微粒群算法是一种随机算法，所以对于步长，我们根据问题的规模在解空间

的某一范围内随机取值）为步长探测适应值更小的点，首先沿着第 1 维的正方向进行探测，如果得到了适应值更好的点，则微粒 x_i 向该维正方向前进 λ，如果沿该维正方向探测得不到更好的点，则沿该维负方向探测，如果得到适应值更好的点，则微粒 x_i 向该维负方向前进 λ，如果仍然得不到更好的点，则微粒在该维上保持不动。然后按照同样的方法沿着微粒的第 2 维，第 3 维……第 n 维进行探测。探测移动完成后，有两种可能的结果，一种结果是经过 n 个维的正反方向的探测，微粒至少移动过一次，到达更好的点 $\hat{x}_i$，有 $f(\hat{x}_i) < f(x_i)$，此时称探测移动成功；另一种结果是沿着所有维的正反方向的探测全部失败，即 $\hat{x}_i = x_i$，此时称为探测移动失败。如果探测移动成功，则执行第二个动作——模式搜索。

3.2.2.2 模式搜索

如果探测移动成功了，则将得到更好的点 $\hat{x}_i$，我们猜测方向 $\hat{x}_i - x_i$ 是一个有利的方向，于是，从当前位置 x_i 出发沿着有利方向跨一步，得到 $M_0 = \hat{x}_i + (\hat{x}_i - x_i) = 2\hat{x}_i - x_i$。这个动作称为模式移动。这一步跨得是否合理呢？我们暂时不急于比较是否有 $f(M_0) < f(\hat{x}_i)$，而是以 M_0 作一次探测移动，得到 M_n，至此完成模式搜索。显然，模式搜索的起点就是前一个动作探测移动的终点 $\hat{x}_i$，而模式搜索的终点是 M_n。如果 $f(M_n) < f(\hat{x}_i)$，则模式搜索成功，$\hat{x}_i \Leftarrow M_n$

当每个微粒的速度和位置得到更新后，以一定的概率对每个微粒进行变异。从探测移动的基本过程来看，对每个微粒 x_i，探测移动要么使微粒移动到更好的位置 $\hat{x}_i$，要么保持在原来的位置 x_i 不动。而当探测移动成功后，即微粒移动到了更好的位置，继而微粒开始执行模式搜索，从模式搜索的过程可知也如同探测移动一样，微粒要么进化，要么不移动。这就是含步长加速变异算子的微粒群算法为什么不需要变异时机的约束，只需要变异概率的限制就可以找到全局最优点的原因所在。即使当微粒聚集到某一局部最优位置时，微粒也可以通过探测移动步长的随机性和模式搜索的鲁棒性跳出局部最优点，只要有一个微粒的位置得到了改进，则聚集的微粒就会自动驱散，进而搜寻更优的位置。

3.2.2.3 算法测试及分析

以 Rastrigin 函数作为测试函数，分别测试带高斯变异的微粒群算法，含维变异算子的微粒群算法，含步长加速变异算子的微粒群算法，取微粒群的种群数 30，维数 10，当 $fun(p_g)<10^{-10}$ 则停止，Station 表示全局最好位置无进化的代数，也就是变异时机参数，变异概率都为 $1/(1+\sqrt{t})$，t 为迭代次数，表 3-3 中各算法的变异时机和变异概率经过测试都能使该算法收敛较快，含步长加速变异算子的微粒群算法的步长 $bc: U(0,3)$，对每种算法测试 10 次，每次测试的收敛代数和 10 次测试的平均收敛代数如表 3-2 和表 3-3。

表 3-2 三种算法 10 次测试的收敛代数

测试次序	群体高斯变异	维变异	步长加速变异
1	1060	1853	1114
2	1248	1411	835
3	3462	1582	775
4	1026	1659	1701
5	1917	1966	1058
6	1351	2005	1778
7	971	2011	884
8	2425	1657	1168
9	1004	1589	916
10	1052	1721	1003

表 3-3 三种变异算法变异时机，变异概率及平均收敛代数

变异类型	变异时机	变异概率	平均收敛代数
群体高斯变异	Station>20	$1/(1+\sqrt{t})$	1551.6
维变异	Station>10	$1/(1+\sqrt{t})$	1745.4
步长加速变异	—	$1/(1+\sqrt{t})$	1119.2

将每种变异算法 10 次测试的收敛代数按照从小到大的顺序排列，作为 10 个坐标点中的纵坐标，横坐标为 1，2…10，画出 10 个点的坐标并画出 10 个点

分段线性插值的曲线，观察各种算法的收敛速度和稳定性，如图 3-2。当横坐标为 i $(i=1,2,\mathrm{L}\ ,10)$ 时，纵坐标分别表示这三种算法 10 次测试中第 i 快的收敛速度，因此这种曲线能够反映这三种变异算法的收敛速度和收敛速度的起伏，即稳定性。

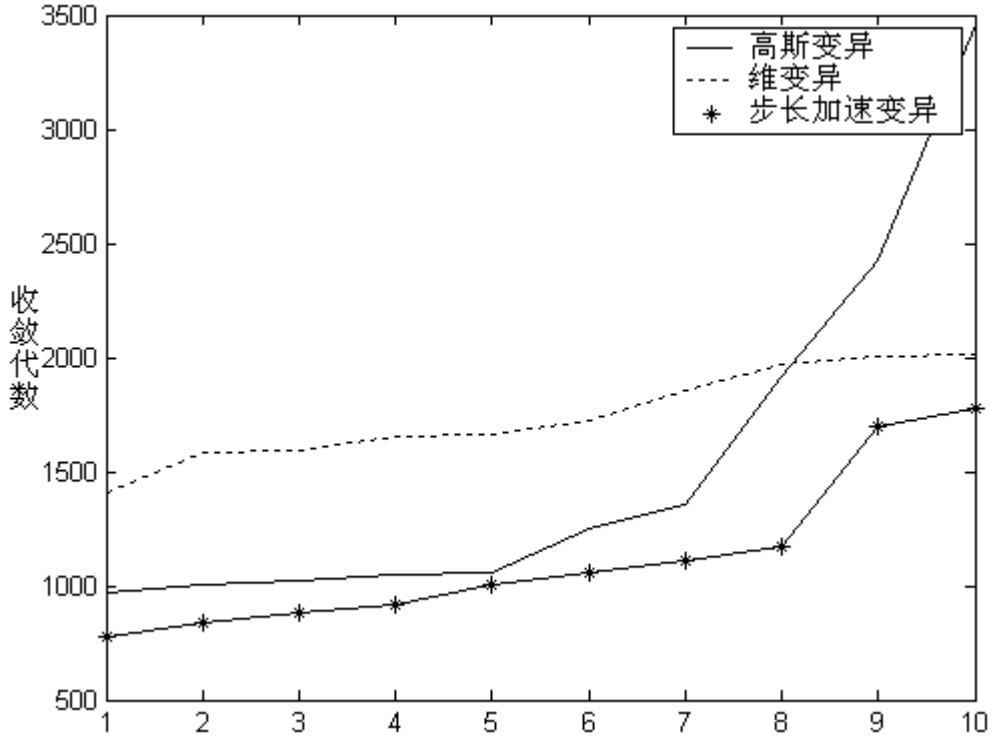

图 3-2 三种变异算法收敛代数变化曲线

从表 3-2 的数据及图 3-2 可以看出带高斯变异的微粒群算法的收敛代数时高时低，显示出了很大的波动性，及稳定性不好；含维变异算子的微粒群算法收敛代数几乎都集中在 1500 到 2000 之间，稳定性虽好，但收敛速度较慢；兼顾稳定性和收敛速度，含步长加速变异算子的微粒群算法效果最好。

3.2.2.4 全局最优位置的步长加速变异微粒群算法

从 3.2.2.2 节的最后一段的分析可以看出，步长加速算子使得微粒 x_i 的适应值得到改进或保持原来的值不变，如果对某一个点 x_1 作多次步长加速迭代，得 $x_2, x_3, x_4\mathrm{L}$ 则 $f(x_1)\geq f(x_2)\geq f(x_3)\geq f(x_4)\geq\mathrm{L}$ 这是一个不增的序列。传统的变异算子都是对每个微粒的位置进行变异，即使是含速度变异算子的微粒群算法，从微粒群算法的位置更新方程 $x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$ 可以看出，微粒速度的变异

最终也不过导致微粒位置的变异，全局最优位置的步长加速变异微粒群算法将是微粒群的全局最优位置做一次步长加速变异，使微粒群的全局最优位置尽可能地保持变化，这样整个微粒群也不至于因全局最优位置处于某个局部最优点时的强大吸引而快速聚集到局部最优点。全局最优位置的步长加速变异微粒群算法不但可以不要变异时机，而且可以不要变异概率，在微粒群的每一次速度和位置得到更新后，直接对微粒的全局最优位置做 1 次步长加速变异，次数过多可能会使全局最优位置偏离各个微粒自身最好位置和每个微粒的当前位置太远，破坏了微粒群的结构。经过测试，微粒群也能很快找到最优点，甚至比微粒位置步长加速变异微粒群算法有更快的收敛速度。

以 Rastrigin 函数作为测试函数，分别测试 1 次和 2 次全局最好位置变异的步长加速变异微粒群算法，取微粒群的种群数 30，维数 10，步长 $bc: U(0,3)$，当 $fun(p_g)<10^{-10}$ 则停止，记录微粒群的收敛代数，对每种算法测试 10 次，每次测试的收敛代数连同表 3-2 中的微粒位置步长加速变异微粒群算法的收敛代数列于表 3-4。

表 3-4 全局最好位置变异和微粒位置变异的步长加速微粒群算法收敛代数

测试次序	1 次全局最好位置变异	2 次全局最好位置变异	步长加速变异
1	1198	662	1114
2	808	1474	835
3	829	1160	775
4	559	1298	1701
5	504	1812	1058
6	746	1980	1778
7	774	2181	884
8	901	1373	1168
9	474	5047	916
10	1390	708	1003

将每种变异算法 10 次测试的收敛代数按照从小到大的顺序排列，作为 10 个坐标点中的纵坐标，横坐标为 1，2…10，画出 10 个点的坐标并画出 10 个点

分段线性插值的曲线，观察各种算法的收敛速度和稳定性，如图 3-3。

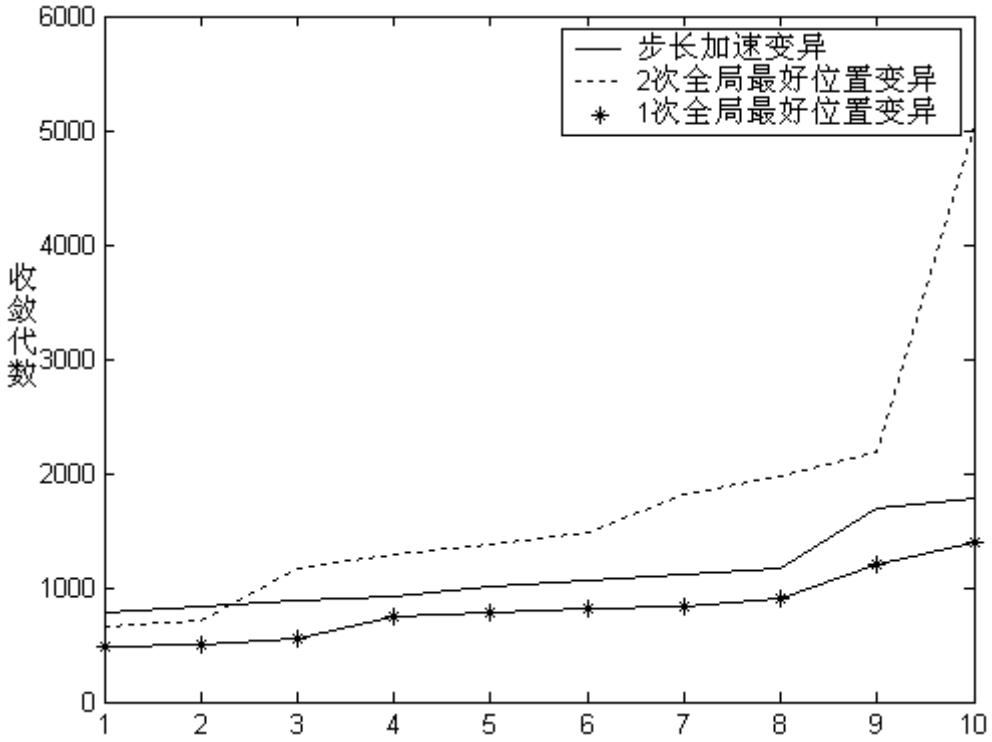

图 3-3 全局最好位置变异和步长加速变异微粒群算法收敛代数变化曲线

从表 3-4 和图 3-3 可以看出 1 次全局最好位置变异的步长加速变异微粒群算法比 2 次全局最好位置变异的步长加速变异微粒群算法有更好的收敛速度和稳定性。全局最好位置变异的步长加速变异微粒群算法比微粒位置步长加速变异微粒群算法有更快的收敛速度。

第四章 微粒群算法在约束优化和整数规划中的应用

在科学与工程领域中，绝大多数的优化问题的求解往往受到各种各样的现实因素的制约，这些制约通常由一系列的约束条件来描述。约束条件可以表示为不等式，等式，数据依赖，数学规划甚至微分方程等。本章提出了一种求解约束优化问题的微粒群算法——保证微粒在可行域内运动的微粒群算法，详细阐述了三种初始微粒群的构造算法和保证微粒在可行域内运动的机理，最后通过约束域为凸集和非凸集两个测试函数的测试，比较说明了三种初始微粒群的构造算法的优劣。

4.1 保证微粒在可行域内运动的微粒群算法

4.1.1 初始微粒群的构造

既然保证微粒在可行域内运动的微粒群算法要始终保持微粒在可行域内运动，那么初始微粒群也必须都在可行域内，如何构造初始可行微粒群呢？以下我们给出三种构造初始可行微粒群的算法。

4.1.1.1 随机压缩半径构造初始可行微粒群

首先找到一个满足可行域的内点 x_0，这一内点 x_0 可以作为第一个初始微粒，然后给出一个足够大的半径 R 保证所产生的微粒有遍历整个可行域的可能性，随机产生一个非零方向 d，d 的每个分量都是(-1，1)上的均匀分布随机数，即 $d = 2*rand(1,n)-1$ (n 为空间的维数)，将方向 d 单位化得 $\overset{1}{d} = d/\|d\|$，作 $z = x_0 + R*\overset{1}{d}$，如果 z 在可行域内，则 z 作为第二个初始微粒；如果 z 不在可行域内，则置 R 为 0 到 R 之间的一个随机数，即 $R = R*rand$，继续判断 $z = x_0 + R*\overset{1}{d}$ 是否在可行域内，如果不满足，则 $R = R*rand$ 直到 z 满足可行域，则 z 作为第二个微粒。然后用同样的方法产生第三个、第四个……直到产生整个微粒群。

下面我们用随机压缩半径构造初始可行微粒群，在约束

$$\begin{cases} x_1 - x_2 - 1 \le 0 \\ x_2 - 1 \le 0 \\ -x_1 - x_2 + 1 \le 0 \end{cases}$$

内产生 1000 个初始微粒，初始内点为 $x_0 = (1, 0.5)$，初始半径 $R = 2$，并画出约束

域边界和微粒群的图形，如图 4-1。

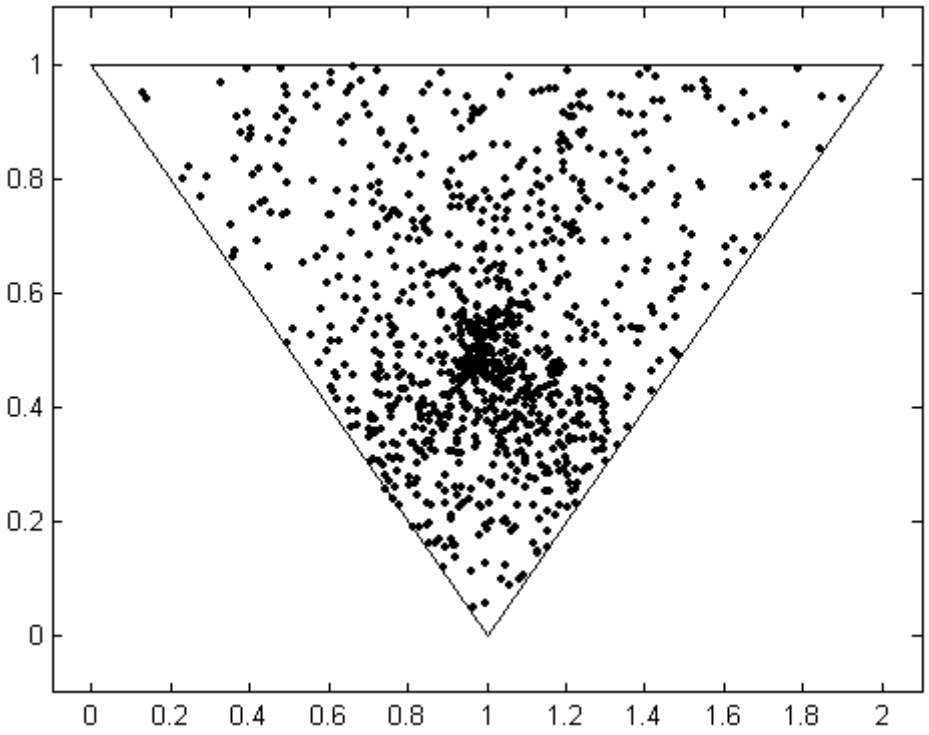

图 4-1 随机压缩半径构造初始可行微粒群

从 4-1 图中可以看出，微粒的分布在初始内点 $x_0 = (1, 0.5)$ 的附近比较密集，在其他区域分布稀疏，在区域的边缘分布最稀疏。

4.1.1.2 比例压缩半径构造初始可行微粒群

由于约束优化问题的最优解一般出现在可行域的边界上，而随机压缩半径构造初始可行微粒群的方法使产生的初始微粒群主要聚集在初始内点的附近，初始微粒群的适应值的状况并不理想，可能会给微粒群的进化造成一定的影响，因此我们引入了比例压缩半径构造初始可行微粒群，其与随机压缩半径构造初始可行微粒群的第一个不同之处在于初始内点不作为第一个微粒，第二个不同之处是当微粒不满足可行域时用 $R = R * \rho$ 压缩半径 R，其中 ρ 称为压缩系数且 $0 < \rho < 1$，为了使产生的初始微粒群尽可能地靠近可行域的边界，ρ 的值应尽量靠近 1。需要指出的是，对于二维问题，我们可以在 $[0, 2\pi]$ 上均匀地产生方向，这样微粒在边界上的分布更趋向均匀。同样用 4.1.1.1 的例子，取初始内点为 $x_0 = (1, 0.5)$，初始半径 $R = 2$，$\rho = 0.99$，随机产生方向，产生 1000 个初始可行微粒群，画出可

行域边界和微粒群，如图 4-2。

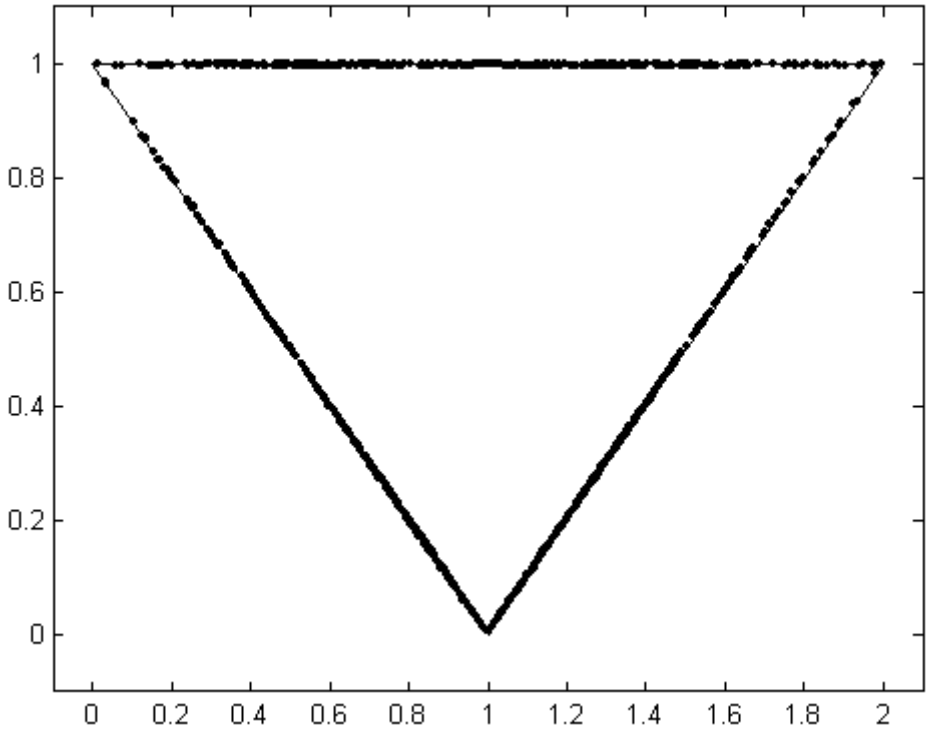

图 4-2 比例压缩半径构造初始可行微粒群

从 4-2 图中可以看出，所有的微粒都靠近可行域的边界，只是有些地方分布不是太均匀，有最优解在分布在较少的微粒边界周围的隐患，对于这个二维问题，我们可以均匀分布方向，则微粒将均匀分布在可行域的边界上，限于篇幅，不再画出均匀分布方向的比例压缩半径构造初始可行微粒群的图形。

4.1.1.3 混合压缩半径构造初始可行微粒群

由图 4-2 可以看出用比例压缩半径构造初始可行微粒群，微粒的分布靠近可行域的边缘，由于约束优化问题的最优解一般分布在可行域的边界上，所以产生的这群微粒之中有靠近最优解的微粒，我们以最接近最优解的微粒作为新的内点估计，同时构造一新的半径 R，用随机压缩半径构造初始可行微粒群，所产生的微粒群中的许多微粒将集中在这个新的内点附近，同时它们也靠近最优解，我们仍然以 4.1.1.1 中的约束为例，其中目标函数为 $\min f(X) = x_1^2 + 4x_2^2$，这个约束优化问题的最优解为 $(0.8, 0.2)$，取初始内点为 $x_0 = (1, 0.5)$，初始半径 $R = 2$，$\rho = 0.95$，随机产生方向，用比例压缩半径构造初始可行微粒群的方法产生 100